Project Proposal

Text Encryption and Decryption

Group Members

Uzair Faraz – (050)

Muhammad Ateeb – (023)

Muhammad Awais – (029)

Malik Shoaib Ali – (047)

Instructor: Ms. Nabeela Bibi

Abstract

This project presents the design and implementation of a menu-driven encryption and decryption system using Assembly Language. The system integrates several classical cryptographic techniques, including Caesar Cipher, XOR-based encryption, Bit Rotation Cipher and Reverse-String encoding. By constructing these algorithms directly through register operations, memory addressing, and low-level instruction flow, the project provides practical insight into how text manipulation and data transformation occur inside the CPU. The user interface relies on Irvine32 library procedures such as ReadString, WriteString, and ReadInt to manage input and output effectively. The final program demonstrates core assembly concepts while offering practical exposure to cryptographic logic and system-level programming.

Introduction

This project focuses on implementing a complete encryption and decryption system in Assembly Language. The program provides a menu-driven interface that allows users to interactively select different cryptographic methods and perform text transformations at the lowest level. By handling data directly through CPU registers and memory, the project strengthens understanding of system programming, low-level operations, and classical cipher mechanisms.

Problem Statement

Modern applications rely on complex encryption frameworks, but understanding how encryption works at a fundamental level requires exploring how characters are manipulated inside the CPU. Assembly Language offers no built-in high-level text manipulation tools, making encryption a challenging task. This project addresses the need for us to understand how classical ciphers function at the instruction level by implementing them manually using registers, loops, conditions, and memory addressing.

Objectives

• Implement multiple classical encryption and decryption algorithms in assembly.
• Build an interactive menu-driven interface.
• Demonstrate register operations, memory addressing, loops, and procedures.
• Utilize interrupts or Irvine32 routines for input and output handling.
• Strengthen understanding of low-level algorithmic processing and string manipulation.
• Provide a comparative understanding of classical ciphers and their logic.

Scope

The system includes several classical encryption algorithms such as Caesar Cipher, XOR Cipher, Bit Rotation Cipher, Reverse-String encoding. Optional inclusion of the Vigenère Cipher depends on instruction-space constraints. The user will be able to enter plaintext, keys, view encrypted results, perform decryption, and repeat operations. Implementation will target x86 Assembly in MASM using Irvine32.

Methodology

1. Requirements Identification

The project begins with determining the constraints of the 8086 environment, selecting feasible encryption techniques, and identifying required registers, memory buffers, and I/O methods.

2. System & Algorithm Design
Data structures, menu flow, and procedure layouts are planned. Each encryption and decryption algorithm is translated into simple register-level operations, and the overall workflow is mapped using flow diagrams.

3. Modular Implementation
The program is implemented using separate procedures for input handling, menu control, encryption, and decryption. Irvine32 routines or interrupts handle user interaction, while loops and register operations perform text transformation.

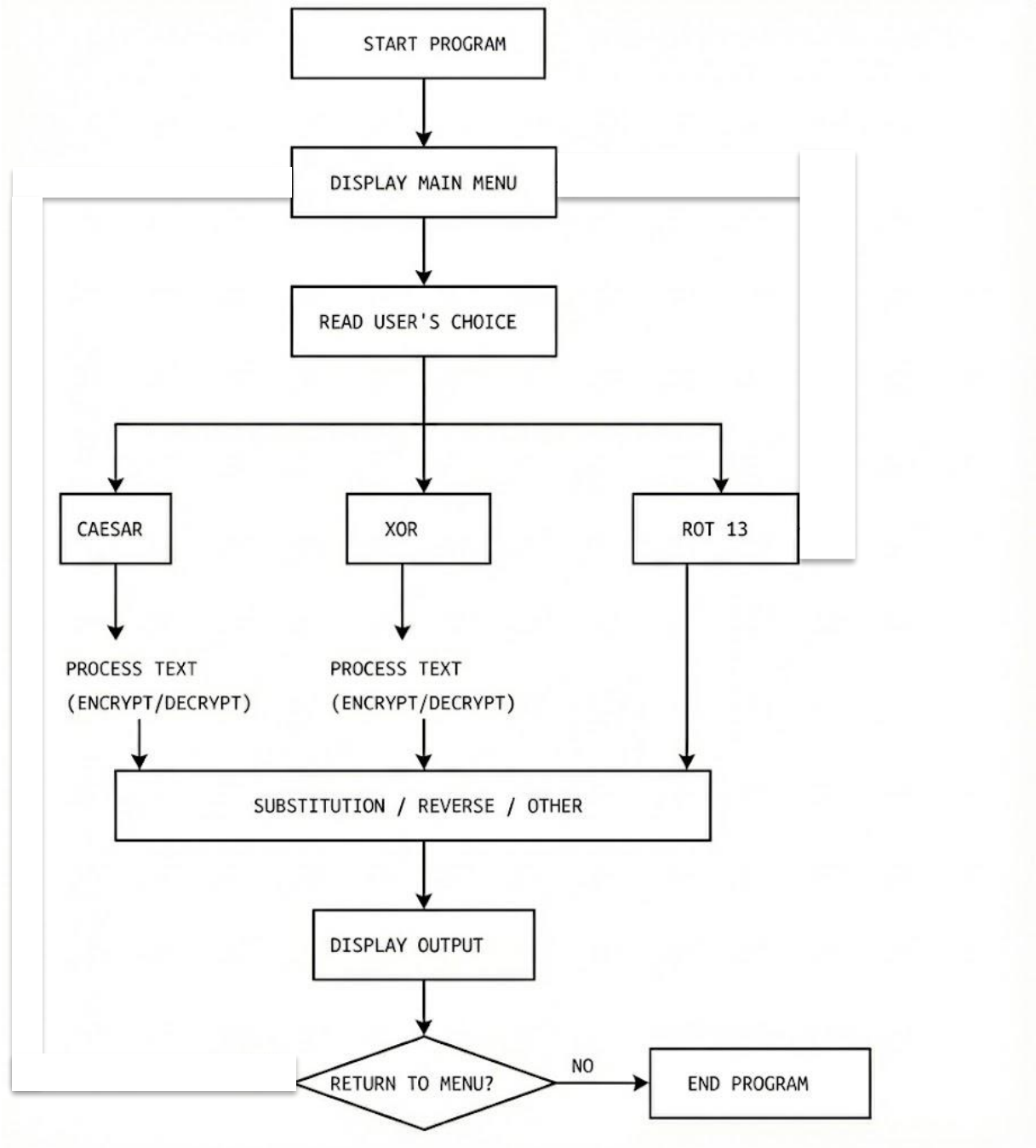
4. Testing & Refinement
Algorithms are tested using sample inputs to verify correctness and reversibility. Edge cases, such as invalid input or boundary conditions, are checked. Minor optimizations ensure efficient register usage and stable program execution.

Workflow

1. Program starts and initializes buffers.
2. Main menu is displayed.
3. User selects an option.
4. User enters text (and key if required).
5. Selected algorithm processes text using register-based logic.
6. Output is displayed.
7. Program returns to menu until the user exits.

Flowchart Design

- Irvine32 Library Functions
- Registers (AL, BL, CL, AX, BX, SI, DI, etc.)
- Arrays / Buffers
- Loops (Simple and Nested)
- Conditional Branching (CMP, JZ, JNZ, etc.)
- Procedures / Modular Functions
- Macros / Constants
- String Handling (Traversal, Input/Output)
- Input Validation
- Menu-Driven Interface
- Interrupts (Optional if DOS INT 21h used)
- Flow Control / Program Workflow
- Data Structures (Arrays for keys or mappings)
- Arithmetic & Logic Operations (Shift, XOR, Add/Subtract)
- Output Formatting (Crlf, WriteChar/String)


START PROGRAM
DISPLAY MAIN MENU
READ USER'S CHOICE
CAESAR
XOR
ROT 13
PROCESS TEXT
(ENCRYPT/DECRYPT)
PROCESS TEXT
(ENCRYPT/DECRYPT)
SUBSTITUTION / REVERSE / OTHER
DISPLAY OUTPUT
RETURN TO MENU?
NO
END PROGRAM

Expected Outcomes

• A complete functional assembly program implementing multiple encryption algorithms.
• Improved understanding of registers, memory access, branching, and loop structures.
• Practical ability to manipulate strings at the machine level.
• Clear demonstration of how classical ciphers operate inside CPU circuitry.

Limitations

• Classical ciphers are not suitable for modern security.
• Input length is limited by fixed buffer allocation.

Conclusion

This project integrates classical cryptography with low-level programming principles. By implementing encryption algorithms directly in Assembly Language, the work builds strong foundational understanding of CPU operations, register logic, and low-level text manipulation. The resulting program is both educational and technically valuable for understanding system-level programming and cryptographic concepts.

THE END